Nombre y Apellido ____________________________________ CI _______________________

Wed Aug 22 20:19:23 2018

En el lenguaje Haskell, ¿a qué se le llama una *definición*?

A la asociación de un nombre (identificador) con su tipo.

A la declaración de una función.

A la firma de una función.

A una asociación de un nombre (identificador) con un valor de un tipo particular.

¿Qué se entiende por expresión en un lenguaje funcional?

Definiciones de funciones cuyos tipos son coherentes.

Cualquier valor que se le puede asignar un nombre.

Combinaciones de operaciones, funciones y valores que evalúan a un resultado.

Cualquier combinación de símbolos que se puede escribir en el lenguaje.

Dada la siguiente función de Haskell:

```
square x = x * x
```

¿cómo se podría evaluar la siguiente expresión?

```
square (3+4) = square 7 = 7 * 7 = 49
```

```
square (3+4) = (3+4) * (3+4) = 7 * (3+4) = 7 * 7 = 49
```

```
square (3+4) = (3+4) * (3+4) = (3+4) * 7 = 7 * 7 = 49
```

Todas las otras opciones son formas válidas de evaluar la expresión dada.

¿Cuál de las expresiones dadas es equivalente a la siguiente expresión de Haskell?

```
sqrt 2 * 2 - 1
```

.

```
((sqrt 2)* 2) - 1
```

```
sqrt (2 * (2 - 1))
```

```
sqrt ((2 * 2) - 1)
```

```
(sqrt (2 * 2)) - 1
```

Considere el siguiente expresión de Haskell:

```
2 * 3 + 4 * 5
```

Indique cual de las siguientes es equivalente.

```
2 * (3 + (4 * 5))
```

```
2 * (3 + 4) * 5
```

```
(2 * 3) + (4 * 5)
```

```
((2 * 3) + 4) * 5
```

Considere el siguiente expresión de *Haskell*:

```
2 * 3 + 4 * 5
```

Indique cuál es el la reducción correcta.

```
2 * 3 + 4 * 5 = 6 + 4 * 5 = 6 + 20 = 26
```

```
2 * 3 + 4 * 5 = 6 + 4 * 5 = 10 * 5 = 50
```

```
2 * 3 + 4 * 5 = 2 * 7 * 5 = 14 * 5 = 70
```

```
2 * (3 + (4 * 5)) = 2 * 3 + 4 * 5 = 2 * 3 + 20 = 2 * 23 = 46
```

¿Cuál de las siguientes condiciones es cierta?

```
"abc" < "abcd"
```

```
"" == ''
```

```
"abcd" < "abcabc"
```

```
"a" == 'a'
```

¿Cuál de las siguientes condiciones es cierta?

```
'False' == "False"
```

```
"True" == True
```

```
False < True
```

```
1 == True
```

Indique cual de las siguientes expresiones de Haskell da 4.

```
4 `mod` (5 `div` 2)
```

```
((-4) `mod` 3) + (8 `div` 2)
```

```
((-4) `mod` 3) + (4 `div` 2)
```

```
(4 `mod` 3) + (5 `div` 2)
```

Considere que || y && son los operadores lógicos de disyunción y conjunción convencionales de Haskell, y que *not* es la negación. Entonces, ¿cuál de las siguientes expresiones evalúa al valor booleano verdadero?

```
not True || False
```

```
 False && not True
```

```
not False && not True
```

```
not False || True
```

Por aplicación de una función se entiende:

Obtener la cantidad de argumentos que tiene.

Chequear si los tipos de los argumentos son los que espera la función.

Dar la definición de una función.

Darle valores de entrada para obtener una salida.

¿Qué entendemos por tipo de dato?

Un conjunto de operaciones y funciones que tienen algún aspecto en común.

Una colección de valores que se consideran juntos porque sobre ellos se pueden aplicar las mismas operaciones.

Una componente de las firmas que se utiliza para resolver la sobrecarga de funciones.

Una marca que se les pone a las operaciones y funciones para poder verificar su coherencia.

¿Cómo se escribe en Haskell el valor booleano *falso*?

`false`

`no`

`0`

`False`

¿Cómo se convierte un valor cualquiera a *String* en Haskell?

En Haskell no existe el tipo de dato *String*.

Con el método *toString()*.

Con la función *show*.

Concatenando el valor con el string vacío ("").

Considere el siguiente codigo Haskell:

```
valor = 2 * 3 + 4 * 5
```

¿Cuál es el tipo de valor?

num

int

Int

number